

More About TAS and IsaWin — Tools for Formal Program Development

Christoph Lüth¹ and Burkhart Wolff²

¹ Bremen Institute of Safe Systems (BISS), FB 3, Universität Bremen
cxl@informatik.uni-bremen.de

² Institut für Informatik, Albert-Ludwigs-Universität Freiburg
wolff@informatik.uni-freiburg.de

Abstract. We present a family of tools for program development and verification, comprising the transformation system TAS and the theorem proving interface IsaWin. Both are based on the theorem prover Isabelle [8], which is used as a generic logical framework here. A graphical user interface, based on the principle of direct manipulation, allows the user to interact with the tool without having to concern himself with the details of the representation within the theorem prover, leaving him to concentrate on the main design decisions of program development or theorem proving.

Introduction

Interactive theorem proving and program development may be more of a challenge than fully automatic techniques, but at the same time it is more powerful. The tools presented here have been designed to make this task as straightforward as possible. They are based on the Isabelle system, which provides a flexible logical framework and combines interactive proof with powerful automatic proof techniques. Isabelle is complemented with a graphical user interface, which is easy to learn and does not burden the user with unnecessary command language syntax and other technicalities. Moreover, the tools and methods can be used for a wide variety of logics or formal methods.

The tools form an integrated system for formal program development, in which TAS is used for transformational program development, and IsaWin for discharging the incurred proof obligations. However, both tools can be used separately as well.

In this extended abstract, we will give a brief overview over TAS and IsaWin. Since TAS and IsaWin have been presented on previous ETAPS conferences [3, 5], the presentation will concentrate on recent improvements.

The Transformation Application System TAS

TAS is a system for formal transformational program development. In a nutshell, formal program development with TAS proceeds as follows. The user begins by

stating an initial specification with respect to a particular signature. The signature, corresponding to an Isabelle theory, is edited in an external file, and contains definitions of operations, data types, axioms, etc. The initial specification is transformed by applying correctness-preserving transformation rules, either generated automatically from existing theorems or taken from a library of predefined transformations. A particular feature of TAS is that transformation rules are based on theorems, which means that we can prove their correctness in a precise logical sense.

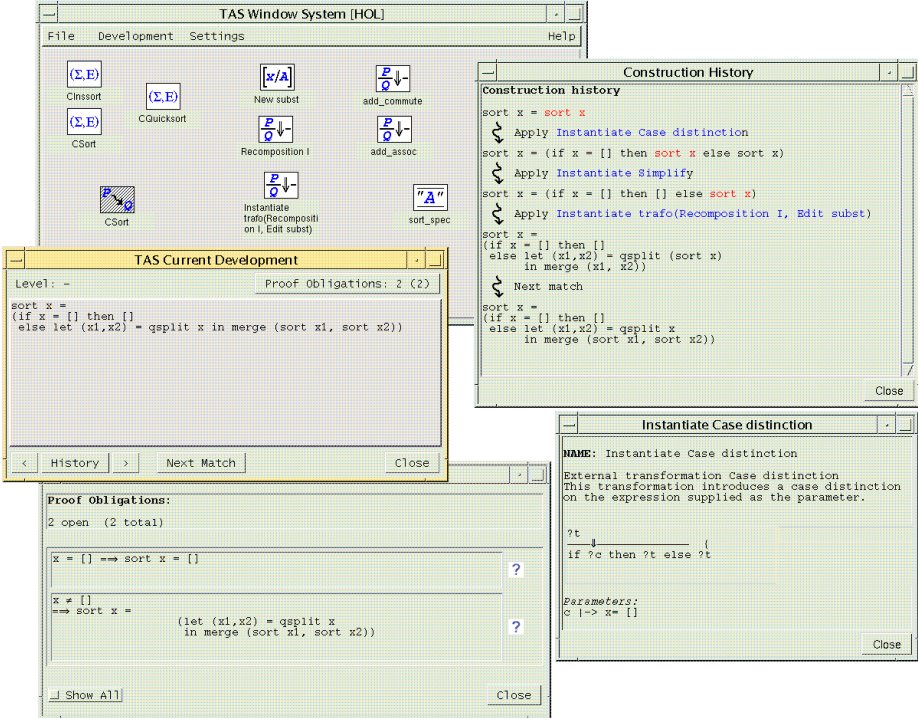


Fig. 1. Screenshot of TAS. Windows, clockwise from top left: the notepad, the construction history, a transformation, the proof obligations and the construction area.

Figure 1 shows a screenshot of TAS. The principle of direct manipulation means that the user should not have to remember the names of transformation rules, theorems, or other objects, but should instead be able to work with meaningful gestures as often as possible. Hence, transformation rules, substitutions and other intermediate results are placed on the *notepad*, where they are represented by icons and can be manipulated by drag&drop. Transformational development takes place in the *construction area*, where the current specification

can be manipulated by *pointing* — we can mark a particular subterm and apply a transformation to it, or ask for all applicable transformations.

TAS is generic over the formal method employed, as long as it supports a notion of *correctness-preserving refinement*. Instantiations of TAS include higher-order logic, where the refinement is based on logical equivalence, and the process calculus CSP, where the refinement is based on the usual refinement of processes.

IsaWin

IsaWin is a graphical user interface for the theorem prover Isabelle. It can be used together with TAS to discharge the proof obligations arising from transformational developments, or as a stand-alone interface to Isabelle. It allows access to all of Isabelle's basic functionality, such as forward and backward resolution, simplification and classical reasoning. An instantiation of IsaWin with the embedding of the algebraic specification language CASL into Isabelle will be presented at the TACAS conference [7].

What's New?

The new developments build on the strengths of TAS and IsaWin: the graphical user interface and its principle of direct manipulation. The basic idea is that we can not relieve users of design decisions during the development or proving process, but we can support them as much as possible. To this end, we attempt to provide only as much information as needed, eliding unnecessary details but preserving the essentials. Details should remain available, but only per user request. The interface should be *quiet* [2]. Of course, users need to be knowledgeable in the particular logic or formal method used, but they should not have to remember names or syntactic representations of transformations, theorems or operations.

The recent improvements can be grouped as follows:

- *Search functions* look for applicable transformations or theorems in a particular situation. The user can mark a subterm, and have the system list all applicable transformations or theorems in a chooser.
- The *transformation library* allows transformations to be grouped into different folders (to distinguish e.g. complex design transformations from simple logical transformations). Further, the system supports the interactive, fully automatic *generation* of transformations from theorems.
- Pervasive use of *interactive hypertext*. E.g. by clicking on the name of an applied transformation rule in the history, the rule itself is displayed; a principle which is used whenever a name is displayed.
- With the *filer*, we can import signatures and specifications from the file system.
- TAS supports the *reuse of developments* by allowing the generation and abstraction of transformation rules from transformational developments.
- A new instantiation of TAS to support a variation of Back and Wright's *refinement calculus* [1] is currently being developed.

More Information

For details of the system architecture underlying TAS and IsaWin, and in particular the generic graphical user interface, we refer to [6]. The wider context of TAS and IsaWin, the UniForM project, is described in [4]. Further information and the tools themselves are available from our web page:

<http://www.informatik.uni-bremen.de/~agbkb/>

System Requirements

To compile TAS and IsaWin, a full Standard ML compiler (such as Standard ML of New Jersey) which furthermore implements the Posix modules from the SML Basis Library is needed. To run TAS and IsaWin, the Tcl/Tk shell (wish, version 8.0 or newer) is needed as well.

Binary distributions are available for Solaris and Linux (SuSE distribution), but the wish needs to be available locally. On these systems, TAS and IsaWin need at least 32 MB to run, and 64 MB to run comfortably.

References

- [1] R.-J. Back and J. von Wright. *Refinement Calculus*. Springer Verlag, 1998.
- [2] R. Bornat and B. Sufrin. Jape's quiet interface. In N. Merriam, editor, *User Interfaces for Theorem Provers UITP '96*, Technical Report, pages 25–34. University of York. Electronic proceedings available at <http://www.cs.york.ac.uk/~nam/uitp96/proceedings.html>, 1996.
- [3] Kolyang, C. Lüth, T. Meier, and B. Wolff. TAS and IsaWin: Generic interfaces for transformational program development and theorem proving. In M. Bidoit and M. Dauchet, editors, *TAPSOFT 97: Theory and Practice of Software Development*, number 1214 in LNCS, pages 855–859. Springer Verlag, 1997.
- [4] B. Krieg-Brückner, J. Peleska, E.-R. Olderog, and A. Baer. The UniForM workbench, a universal development environment for formal methods. In J. M. Wing, J. Woodcock, and J. Davies, editors, *FM'99 — Formal Methods. Proceedings, Vol. II*, number 1709 in LNCS, pages 1186–1205. Springer Verlag, 1999.
- [5] C. Lüth, H. Tej, Kolyang, and B. Krieg-Brückner. TAS and IsaWin: Tools for transformational program development and theorem proving. In J.-P. Finance, editor, *Fundamental Approaches to Software Engineering FASE'99. Joint European Conferences on Theory and Practice of Software ETAPS'99*, number 1577 in LNCS, pages 239–243. Springer Verlag, 1999.
- [6] C. Lüth and B. Wolff. Functional design and implementation of graphical user interfaces for theorem provers. *Journal of Functional Programming*, 9(2):167–189, March 1999.
- [7] T. Mossakowski. CASL — from semantics to tools. In *TACAS 2000 — Tools and Algorithms for the Construction and Analysis of Systems*. Springer Verlag, 2000. To appear in LNCS.
- [8] L. C. Paulson. *Isabelle - A Generic Theorem Prover*. Number 828 in LNCS. Springer Verlag, 1994.